> skilled in the art to make and use the present invention. Without a computer listing, the present invention amounts to merely to a conceptual outline of the present invention, and descriptive material absent of a clear teaching on how to make and use the invention.

The examiner thus articulated a per se rule that source code was needed to establish enablement. Applicants disagree with this rejection and contend that the examiner is incorrect in terms of the legal basis, and also based on a comparison to applications for comparable subject matter.

First of all, to say that "without on a computer listing, the present invention amounts merely to a conceptual outline," is analogous to saying that a patent to a mechanical device that shows the building blocks must be incomplete without a list of specific parts and dimensions of materials. The courts have recognized that "when disclosure of software is required, it is generally sufficient if the functions of the software are disclosed, it usually being the case that creation of the specific source code is within the skill of the art." Robotic Vision Sys., Inc. v. View Eng'g., Inc. (Fed. Cir. 1999) (emphasis added). The examiner has not indicated how this case is different from what is "usually [] the case."

The level of disclosure in the present application is typical of applications filed in this technology at this time. For example, the examiner has cited against the claims international publication number WO 98/44695 to Apple Computer. This application issued on June 26, 2001 as U.S. Patent No. 6,253,228. This document was internationally filed in March, 1998 and published in October, 1998, two months before the present application was filed. In that document, there are seven figures, of which Fig. 1 is a prior art overview of a computer; Figs. 2, 3, 4, and 7 are simple figures showing relationships between an Apple and a controller and simple associational bindings, and Figs. 5, 6A, and 6B are flow charts that mimic language in the specification. The publication also has tables with short pieces of code. The present application, while having a little less description, is comparably described in terms of block diagrams, flow charts, and short examples of code. The Apple document thus demonstrates that such level of disclosure is typical and allowable in such applications.

Furthermore, if the examiner continues to maintain the rejection under § 112 on the basis that an application directed to software is not enabled if source code is not provided, the examiner should withdraw the rejection on the basis of the Apple document, because then that document would be for a nonenabling disclosure.

Applicants thus contend that the examiner has not applied a reasonable basis for this rejection, that the basis is based on an incorrect per se rule relating to the disclosure of source code, and that the disclosure is consistent with comparable applications.

The examiner separately rejected claims 5 and 7 under § 112, first paragraph. The examiner contends that "there is insufficient evidence to suggest that applicants had de facto reduced his invention in both descriptive language and a compiled language." There is no requirement for an actual reduction to practice. In addition, it is known that there are many languages, some of which are compiled, and other interpreted. See, for example, the Apple publication, page 18, regarding references to interpreted scripts and to Objective C and C++ (which are compiled).

III. Rejections Over Apple.

Claims 1-3, 5-9, 12, 13, and 16 were rejected under § 102 as anticipated by WO 98/44695 ("Apple"). The examiner contended that the limitations of independent claim 1 were shown in Apply by action bindings 402 and the keys bound to specific objects or variables in the server.

Claims 4, 10, 11, 14, 15, and 17 were rejected under § 103(a) over Apple.

Apple relates to a method and apparatus for updating and synchronizing information between a client browser and a web server. The application states that it was known to enter input data at a browser and to process that data at a web server in a stateless manner such that changes to the data would require a user to start over with an input form. To do this, the web browser has an applet group controller, and the server transmits identification numbers of applets listed on a page along with the page. User-entered values create class instances at the web browser. These applets can then be used to synchronize information with the web server, and allow parts of a page to be updated without re-rendering the entire page. These applets are said to take the place of a FORM tag.

As indicated in Table 1, a WEBOBJECT tag has a name for a particular field or button, and that name corresponds to an entry in a declarations file. The INPUTFIELD instance of the applet has an association key (which in Table 2 is the same for all three fields) and a value or action key. The word INPUTFIELD is used to identify the INPUTFIELD in the declaration file. Consequently, it appears that if INPUTFIELD is misspelled in the HTML code as shown in Table 1 in Apple, the link to that declaration will not be made.

In the present claims, the input field is mapped to a property on rendering, and thus the system can overcome misspellings or incorrect naming in the input field name, as described at pages 2-3 and 10 of the application.

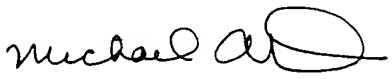
Claims 1, 12, and 16 have been amended. Most of the changes here are intended to be clarifications, to make for better reading, and to provide antecedent basis for other claims. It is believed that most of these changes to do not affect substance of the claim and do not amount to a narrowing of the claim. The claims have also been amended to state that the mapping is done such that the software component can process the mapped input data even with an incorrectly spelled or named input field name in the hypertext document. As described in the application, this ability can help avoid naming users. As indicated above, Apple does not teach or suggest this feature of claims 1, 12, or 16.

IV. Conclusion.

All claims should be in condition for allowance, and accordingly a notice of allowance is respectfully requested.

Please charge any fee in connection with this matter to our Deposit Account No. 08.0219.

Respectfully submitted,



Michael A. Diener
Attorney for Applicants

Hale and Dorr LLP
60 State Street
Boston, MA 02109
October 22, 2001
Attorney Docket No. 101.957.156

CLAIMS AS AMENDED

1. (Amended) A method comprising:

rendering a hypertext document including emitting program code for mapping input field names in the hypertext document to software component properties when the hypertext document is rendered, the software component being a server-based component that uses data from the input field for processing;

providing the rendered hypertext document to a user;

receiving from the user input field data entered in a named input field;

determining from the mapping a software component property mapped to the named input field; and

calling the software component for processing the input field data, the mapping being done such that the software component can process the input field data regardless of the spelling of the name of the input field in the hypertext document.

2. The method of claim 1, wherein the rendering includes emitting hypertext form tags with a current value of an input field pre-filled in.

3. The method of claim 1, wherein the mapping includes encoding the hypertext input form with a unique name and registering the name.

4. The method of claim 3, wherein the receiving includes determining if the user submitted input field data is from a hypertext input form and bypassing input field processing if the determination cannot be made.

5. The method of claim 1, wherein the program code is source code of a compiled programming language.

6. The method of claim 1, wherein the program code is source code of an interpreted programming language.

7. The method of claim 1, wherein the program code is object code of a compiled programming language.

8. The method of claim 1, wherein the program code is object code of an interpreted programming language.

9. The method of claim 1, further comprising converting the submitted input field data to a correct data type.

10. The method of claim 1, wherein the determining includes iteratively processing input names associated with a component property to determine if data associated with any of the input names has been entered.

11. The method of claim 10, wherein the determining includes processing in order of priority stored with the mapping of the input field names.

12. (Amended) A web server system comprising:

a preprocessor for generating program code to register mappings between hypertext input field names and software component properties and to emit hypertext form tags, the software component being a server-based component that uses data from the input field for processing;

a name-space manager for registering the mappings; and

a data handler, responsive to input data submitted by a user with an input field name, for using the mappings in the name-space manager to associate the input field names with the appropriate component properties, and for calling the software component to process the input data, the mapping allowing the software component to process the entered data regardless of the name of the input field in the hypertext document.

13. The system of claim 12, wherein the data handler converts the submitted input data to a correct data type.

14. The system of claim 12, wherein the name-space manager includes a table for mapping a form to input fields, and for mapping input fields to a component property.

15. The system of claim 12, wherein the name-space manager includes a table for mapping a form to input fields, and for mapping input fields to a priority that determines the order in which the data handler processes the input field mappings.

16. (Amended) A method comprising processing a hypertext document by identifying and storing input field names before sending the document to a user, and in response to receiving input data with input field names from the user, determining appropriate software component methods for processing the input data by looking to the stored input field names, the input names for the input data being determined from the document without requiring that the input names be correctly named.

17. The method of claim 16, wherein the storing includes storing a priority for each input name.

CLAIMS AS ~~FILED~~ AMENDED

1. **(Amended)** A method ~~for mapping input fields in~~ **comprising:**

rendering a hypertext document **including emitting program code for** ~~to software components, the method comprising:~~

mapping input field names in ~~a~~ **the** hypertext document to **software** component properties when the hypertext document is rendered**, the software component being a server-based component that uses data from the input field for processing**;

providing the rendered hypertext document to a user;

receiving from the user input field data **entered** in a named input field;

~~and~~

~~using~~ **determining from** the mapping ~~to determine an appropriate~~ **a software** component property ~~for~~ **mapped to** the named input field**;** and

~~to call~~ **calling the software** component ~~methods~~ for processing the input field data**, the mapping being done such that the software component can process the input field data regardless of the spelling of the name of the input field in the hypertext document**.

2. The method of claim 1, wherein the rendering includes emitting hypertext form tags with a current value of an input field pre-filled in.

3. The method of claim 1, wherein the mapping includes encoding the hypertext input form with a unique name and registering the name.

4. The method of claim 3, wherein the receiving includes determining if the user submitted input field data is from a hypertext input form and bypassing input field processing if the determination cannot be made.

5. The method of claim 1, wherein the program code is source code of a compiled programming language.

6. The method of claim 1, wherein the program code is source code of an interpreted programming language.

7. The method of claim 1, wherein the program code is object code of a compiled programming language.

8. The method of claim 1, wherein the program code is object code of an interpreted programming language.

9. The method of claim 1, further comprising converting the submitted input field data to a correct data type.

10. The method of claim 1, wherein the determining includes iteratively processing input names associated with a component property to determine if data associated with any of the input names has been entered.

11. The method of claim 10, wherein the determining includes processing in order of priority stored with the mapping of the input field names.

12. ~~A system for mapping hypertext input fields to software components~~**(Amended) A web server system** comprising:

a preprocessor for generating program code to register mappings between hypertext input field names and **software** component properties and to emit hypertext form tags**, the software component being a server-based component that uses data from the input field for processing**;

a name-space manager for registering the mappings ~~and rendering the document so that the document can be provided to a user~~; and

a data handler, responsive to ~~submitted~~ input data **submitted by a user** with an input field name ~~submitted by a user~~, for using the mappings **in the name-space manager** to associate **the** input field names with **the appropriate** component properties, and for calling ~~appropriate component methods for processing the input data.~~ **the software component to process the**

input data, the mapping allowing the software component to process the entered data regardless of the name of the input field in the hypertext document.

13. The system of claim 12, wherein the data handler converts the submitted input data to a correct data type.

14. The system of claim 12, wherein the name-space manager includes a table for mapping a form to input fields, and for mapping input fields to a component property.

15. The system of claim 12, wherein the name-space manager includes a table for mapping a form to input fields, and for mapping input fields to a priority that determines the order in which the data handler processes the input field mappings.

16. **(Amended)** A method comprising processing a hypertext document by identifying and storing input field names before sending the document to a user, and in response to receiving input data with input field names from the user, determining appropriate software component methods for processing the input data by looking to the stored input field names, ~~whereby~~ the input names for the input data ~~are~~ **being** determined ~~form~~ **from** the document without requiring that the ~~names be separately provided in the document and in the software component.~~ **input names be correctly named.**

17. The method of claim 16, wherein the storing includes storing a priority for each input name.